4-10-2016

Uncertainty Avoidance—A New Teaching/ Learning Method for an Introductory Programming Course

Zhen Jiang
West Chester University of Pennsylvania, zjiang@wcupa.edu

文章编号(Article Code)：1672-5913(2016)04-0052-07 中图分类号(CLC)：G642

Uncertainty Avoidance—A New Teaching/ Learning Method for an Introductory Programming Course

Zhen Jiang

(Department of Computer Science, Information Security Center, West Chester University, West Chester, PA 19383, USA)

Abstract: In this paper, we introduce a new procedure for under-represented students to quickly learn the use of the decision structure in computer programming. The challenge here is to help students, who lack sufficient background of mathematics and computer programming, to use this structure correctly without too much doubt and uncertainty. The traditional CS0 program elapses several semesters and requires many foundation courses to be taken before the students have knowledge of the program correctness. Our one-semester course CSC115 allows students to build up programming skills gradually case by case and program by program. Such a guideline is proven to be effective for those inexperienced students to write correct If-else/If statements without the need for learning formal methods in classroom.

Key words: certificated course curriculum; education module and experience; uncertainty and doubt in introductory computer education.

1 Introduction

Cyber security is a hot topic. A number of institutes designated by the National Security Agency (NSA)[1] has provided a high standard (e.g., Ref. [2]) of education in this area. One of the NSA certified programs[3] at West Chester University(WCU) is set to engage under-represented non-CS（computer science）majors in this interesting and important issue.

To attract more undergraduates from a broad range of fields to our rigorous introductory course, a problem cannot be ignore is the students' feeling of uncertainty and doubt. Due to the lack of sufficient background, those students may encounter more obstacles than CS major students to hurdle in the class. Any unexpected challenge, as those students called it "the uncertain thing," can scare them away from a full dedication to the learning, causing an inefficient learning or even a total quit.

To completely solve the problem, we need an efficient pedagogy method to help students learn the materials with a significant technical depth in a very short time frame, paving the road for inexperience students to do what a CS graduate may have in that class. By the time we give them an explanation with sufficient technical background on what they concern about, we also need to ensure that they can fully understand it without any obstacle.

In this paper, we demonstrate our practice in an introductory programming class, in attempting to solve the aforementioned uncertainty problem. We first show a very common scenario that raises the students' uncertainty, when we introduce the decision structure in such an introductory course. Then we provide our solution. From our practice, this approach helps the target students to build a complex program quickly

in a convenient and error-free way. By continuously working and making progress under our guidance, more non-CS majors reach our education goal as a CS graduate can do in using the decision structure.

The remainder is organized as follows: Section 2 describes the target problem. Section 3 claims the idea of our solution. In section 4, our practice is introduced in details. Section 5 summarizes the results. Section 6 presents some conclusions.

2 Problem

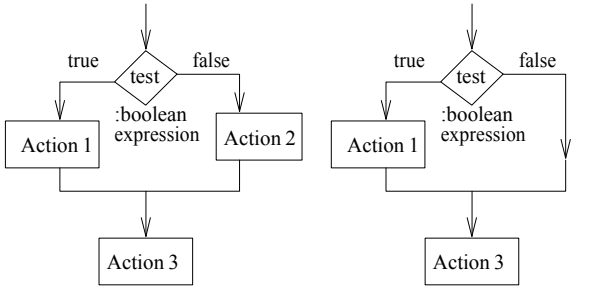
"CSC115 - Introduction to Computer Programming" is an entry-level programming course[4] and is the only prerequisite course for non-computer-science majors to take our NSA- certified topic courses[5]. This course aims to structure a programming basis for the development of complex computer application in the succeeding topic courses. It is for all non-CS majors, with up to 8 sessions each semester, four times the number of sessions of the entry- level programming course that we make available to CS major.

The if-else statement is a basic decision structure introduced in this VB course (see Fig. 1 (a)). However, our non-CS major students may not have sufficient experience to interpret their understanding of requirements with the program statements. A simple development of the boolean expression such as the check of a leap year can scare them away and make them easily drop the class in the middle of semester. Usually, this problem has occurred early in their first if-else program when they do not have sufficient knowledge (e.g., [6]) to ensure a correct boolean expression in the test part.

For instance, when a program to check the passing score (i.e., an integer read from a text field, 60 or above) is discussed in class, students may give different answers as shown in Fig. 1 (b). If this ambiguity cannot be explained clearly in time, its snowball effect may create too much tension and make our students doubt on their programming skills and capability.

3 Idea

In the normal learning process in the CS0 program [7], the



(a)

Dim grade as Integer = Input.text
If grade > 59 then
Output.text = "Congratulations"
' print process
end if
	Dim grade as Integer = Input.text
	If grade >= 60 then
	Output.text = "Congratulations"
	' print process
	end if

(b)

Fig. 1 (a) Syntax of If-else and If statements, and (b) Uncertain development of the boolean expression.

students will gain programming skills by accumulating experience from a lot of practices in order to build up the knowledge of the concepts and styles. To develop the code correctly, students need to learn advanced concept and techniques such as the operational semantics. Our CSC115 course has a strict time constraint in one semester. Therefore, a practical method is needed to effectively guarantee the quality of programming, while those advanced concepts and techniques must be transparent.

In the following, we introduce our practice in a five-phases learning process, as well as the corresponding assessment plan. Step by step, students can obtain all they needed to develop a correct program on the decision making and solve the above uncertainty problem, because the obstacles can be mitigated in their gradual progresses.

4 Uncertainty a Voidance in Our Education Practice

After basic concepts and flow chart of the If-else statement are introduced in class, the students in our class will walk through five phases to obtain the required skills and knowledge. These five phases form a spiral learning model, with each step assessed easily. More importantly, this model interprets the entire

development process. For any mistake or missing part students made in the design, coding, or testing, the instructor can quickly locate the training part that needs to be reinforced and help students to hurdle the obstacles (i.e., uncertainty avoidance).

4.1 Boolean expression, complex one, and their values

In this phase, the students need to learn the format and the evaluation of a boolean expression (in Visual Basic). The common issue made by those non-CS major students is the chained-comparison-operators such as $2 < a < 4$.

We start from the simple expression with 2 values and 1 (relational) operator only. Then, the complex one consists of a number of simple expressions that are connected with "AND", "OR", and/or "NOT". After the precedence order is introduced, a student should be able to evaluate an expression like "NOT $a > 2$ OR $a < 7$ AND $a > 3 + 2 \times (-2 + 3)$" (by given a value of variable a).

4.2 Tracing of If-else and If statements

In this phase, the students need to know the result from a given program with the If-else and/or If statements. The confusion might by raised when the curly braces are omitted due to the use of the single line command in action(s).

We test our students with the nested If-else/If structure. Each test point is encoded in the binary format in bits, say with a label 1, 2, 4, 8,⋯ then, an addition is applied at each place, just simply adding such a label to the test variable. After that, each addition at the precise moment of execution time can be read from the final value of this variable. The computer's result will be compared with the student analysis, in order to help student to ensure the reaching and the sequence of these test points in the execution. An example of this kind of testing program can be seen in the following.

```
x = 90
y = 0
if x < 60 then
y = y + 1
if x > 80 then
y = y + 2                                    (1)
else
y = y + 4
end if
end if
listbox1.Items.add(y)
```

To help the student successfully reach this goal, we start from the tracing of a single If-else or If statement. Before we test them with the above addition operations, we can have a warming up test. Instead of using the addition, an assignment is adopted, which relies on the final value update only. That is, replace each "y = y +" by "y=" in the above.

4.3 What is that missing (relational) operator?

From this phase, we start to build up students' programming skills. We ask student to give the right operator in a simple boolean expression for the test part in a single If-else/If statement.

We provide students with a five-steps development pro- cedure. The details can be seen in Algorithm 1.

Algorithm 1: Decision structure development.

(1) Determine whether it is a problem with 2 exclusive cases only (i.e., either or but not both). So, the problem can be solved with a single If-else/If statement.

(2) Implement each case in different action parts.

(3) Identify the situation values for each case selection.

(4) Make a boolean expression so that all situation values for Action 1 will lead to evaluation value true.

(5) Verify whether all situation values for Action 2 will lead to the evaluation value false. Otherwise, go back the above step 4.

In step 1, the target problem is ensured to solve with the syntax of If-else/If statement in Fig. 1 (a). In step 2, two actions are implemented, one for each case. In step 3, the condition to select each case is interpreted with the values that can be accepted by the computer. Then, in step 4, the boolean expression is designed. It aims to a true field that contains all the options to Action 1 (according to the situation values). In step 5, a false field that contains all the options to Action 2 is verified with the boolean expression. Once the boolean

expression passes the checks in both steps 4 and 5, its functionality as the test part in the If-else/If statement can be guaranteed correct.

For the above program in Fig. 1 (b), we can ensure that the exclusive pass/fail can be developed into a single If-else/If statement. The action of each case, such as screen display "Congratulations" for those students who pass the test, can be implemented. It is easy to determine the situation values for the passed case when the student "grade" is 60, 61, … , or 100. In step 4 of Alg. 1, "grade>= 59," "grade> 59," "grade<> 59," and "grade>= 60" will pass the check. However, only two of them, "grade> 59" and "grade>= 60" pass the check in step 5 (of Alg. 1), when the situation values for the failed case (i.e., "grade") is 59, 58, … , or 0. Either one of them can be used as the required expression in the test part. Therefore, students can avoid confusion and ensure the code correct. That is, the aforementioned uncertainty can be mitigated. Note that our approach is practical since those advanced technologies in proving the correctness of code have been transparent to students now. But it opens a window for them to study those advanced materials later if they are really interested in this algorithm itself.

In this phase, students' learning with this algorithm is assessed in their work, by finding the missing relational operator to complete a simple boolean expression. For example, students will be asked to give the identical code that prints out the same result as the sample program in Fig. 1 (b) does. This becomes a complete test after: ① the use of threshold value from true field (i.e., 60) vs. the one from false field (i.e., 59), ② a switch of variable and threshold value (e.g., "grade> 59" vs. "59 <grade"), ③ a switch of actions 1 and 2 (pass/fail implementations), and ④ the use of the complement of test condition (simply adding "NOT" ahead, e.g., "NOT(grade<= 59)" vs. "grade> 59"), in total $2 \times 2 \times 2 \times 2 = 16$ different formats.

4.4 Development of the entire boolean expression

In this phase, students are asked to complete the test part of a single If-else/If statement with Algorithm 1, either in the simple format or the complex one.

For each value checked in a single expression, the true field and the false field will be identified (in step 3 of Algorithm 1). Then, the critical value will be selected as the threshold in the expression. For instance, 60/59 is used in the check of pass/fail, and 0/1 is used in the check of positive/not positive. For more than one value checked in a complex expression, the true field needs a calculation with the intersection ("AND") and/or the union ("OR"), or even an intersection distributed over the union "A AND (B OR C)" and/or a union distributed over the intersection "A OR (B AND C)." The false field will need a calculation with the complement (of the true field) to ensure the coverage of options to Action 2.

For instance, a leap year check requires the boolean expression "year MOD 100 <> 0 AND year MOD 4 = 0 OR year MOD 400 = 0," where MOD is the modulo operator to get the remainder of the integer division. This is a union distributed over the intersection.

In a simple word, if we find a range in the true field for a checked value t, say $t \in [x, y]$, the expression can be interpreted with an intersection of the selections of both bounds x and y, say "$x \leqslant t$ AND $t \leqslant y$." If the true field is a union of two separated ranges, a union OR is needed for a conjunct expression with the ones to check the corresponding separated ranges in their true field. This kind of selection on true field can be applied in multiple layers, until approaching the target (true) field.

For instance, to identify a given integer t as an even number in the range from –100 to 100, but not divisible by 3 such as 6, an expression can be written in an intersection conjunction with three layers: "t MOD 2 = 0," "$-100 \leqslant t$ AND $t \leqslant 100$," and "t MOD 3 <> 0."

4.5 Multiple case problem is solved with If-else/If statements

In this phase, students are required to extend the use of If-else/If statement from a T/F case problem to the multiple case problem.

We provide a general guidance for the students to use the nested structure. The details are shown in Algorithm 2.

Algorithm 2: Guidance for the decision structure development.

(1) Determine the program is a singleton-, 2-, or multi-case problem.

(2) Use If statement (of If-else statement) to solve the singleton-case problem (or the 2-case problem) with Alg. 1. Then, end the entire process.

(3) Otherwise, apply the above step 2 to solve the first case in the multi-case problem and then repeat step 1 for the rest cases.

The idea is straightforward. Simply, the situation in the easiest selection is chosen as the target first. Its true field will be used to determine the test part in the exterior If-else statement. Then the rest of problem becomes one case less and will be implemented in the false field. This process will continue until the problem can be simplified into a 2-case problem and solved in one single If-else statement with Algorithm 1.

For example, the problem: "15% tip of a meal, with the minimum $1, but cannot exceed the amount of the meal price itself" has an easy case when tip>$1. Then, the rest of problem can be implemented in another If-else statement (which becomes the interior), by checking whether a tip of $1 is over the meal price or not. The resultant program can be seen in the following.

```
Tip = meal * .15
if tip > 1 then
listbox1.Items.add(tip) ' 15%
else
   if meal > 1 then                                    (2)
   listbox1.items.add(1) ' $1
   else
   listbox1.Items.add(meal) ' < $1
   end if
end if
```

Fig. 2 shows the overall picture of development for all kinds of decision problems (i.e., singleton-, 2-, and multi-case problems).

5 Result from Class Observation

5.1 Rationales and consequences at the technical level

After students walked through the above 5 phases in

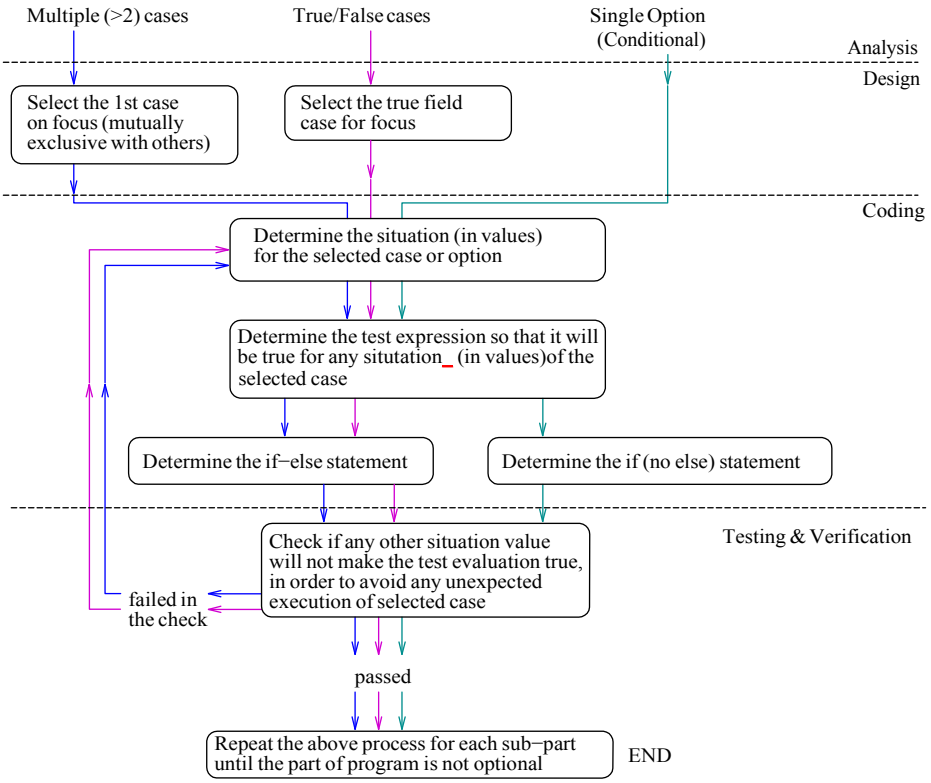


Fig. 2 Activity diagram of programming with the If-else/If statement.

Call Numbers	Location
100 to 199	basement
200 to 500 and over 900	main floor
501 to 900 except 700 to 750	upper floor
700 to 750	archives

```
if num >= 100 and num <=199 then
  Output.text = "basement"
else
  if num >=200 and num <= 500 or num > 900 then
    Output.text = "main floor"
  else
    if (num >= 501 and num <700) or (num > 750 and num <=900) the
      Output.text = "upper floor"
    else
      Output.text = "archives"
    end if
  end if
end if
```

Fig. 3 A sample VB development for multiple case problem.

classes, they obtain the skills and knowledge to solve those challenging problems left in the textbook. For instance, Fig. 3 shows an example of the multiple selection program [8]. It is to print out the location of books by given the call numbers. The development will be initiated along the blue line in Fig. 2.

The condition with its situation values (∈ [100, 199]) is identified for the first case in which "basement" will be printed out. After the boolean expression is determined in step 3 of Algorithem 2, the development will focus on the rest of the cases. This process (highlighted in blue line in Fig. 2) will be repeated until the entire problem is left in two cases: "upper floor" and "archives." Then, the problem is solved with the procedure along the purple line in Fig. 2. At the end, the final program is created.

From the guidance in Fig. 2, our students have the clue to start their work in solving the problem. They will not feel uncontrollable or uncertain any more. From a test with questions in the aforementioned levels, non-CS major students can achieve an 87% success rate as the level we assessed in another class [9] for major students in the ABET program. Compared with the student outcomes before this training, the student success rate is enhanced by up to 129% (see this promotion from 38% to 87% in Fig. 4). This enhancement helps WCU to prepare and promote more mature workforce to the advanced topic courses in the computer area.

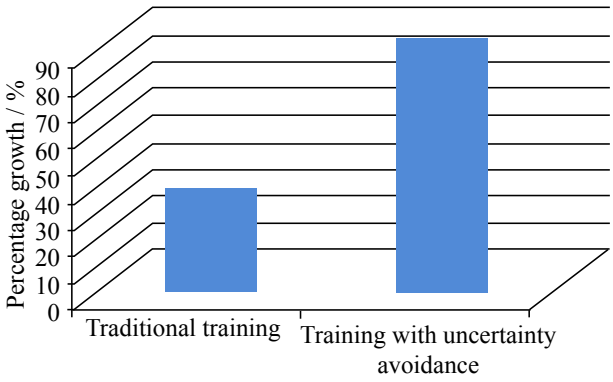


Fig. 4 Student performance in different training processes.

5.2 Rationales and consequences at the pedagogical level

Our goal is to find a practical method and help students walk out from the uncertainty problem. As demonstrated here, by following our guidance, students can easily obtain the skills and knowledge needed for the decision structure development. Our practice also shows that this training is very quick. For non-CS majors, it may take only 2 weeks to wrap up everything in the chapter of this decision making. When it is extended to the training for CS majors, this work can be done in one week [9].

With the enhancement of student success shown in Fig. 4, our general education in CSC115 can hold a pretty high retention rate of non-CS majors in our computer classes, avoiding students dropping their training opportunity in the middle.

6 Conclusion

The specified uncertainty problem in teaching trainees who have little experience with the program correctness is of intrinsic interest because of its economic importance and potential market value. It is clear that computer science certificate programs are used in many places. These programs vary by language, application, and teaching method.

In this paper, we have shown our training with the uncertainty avoidance. This guides the students to build programs with the If-else/If statements. This guidance also very effectively helps to ensure the program correctness and speeds up the development, especially for those programmers who lack enough background of the formal methods. Its reuse is not only limited to the trainees to develop different programs, but also extended to trainers who will like to repeat the success of programmer training.

Our search did not yield any complete comparison. Based on the discussion above, it is clear that learning programming easily and quickly with the use of this proposed education model works better with practice.

References

[1] National Centers of Academic Excellence in Information Assurance (IA)/Cyber Defense (CD)[EB/OL]. [2016-01-18]. https://www.nsa.gov/ia/academic_outreach/nat_cae/.
[2] National Security Telecommunications and Information System Security (NSTISS). National Training Standard for Information Systems Security[EB/OL]. [2016-01-18]. (1994-06-20).http://www.ecs.csus.edu/csc/iac/4011.pdf.
[3] CSC110 & CSC115, General Education Courses for non-CS Majors, Computer Science Department, West Chester University[EB/OL]. [2016-01-18]. http://www.wcupa.edu/-information/official.documents/undergrad.Catalog/compsci.htm.
[4] CSC115-Introducation to Computer Programming, special session of Matlab for non-CS majors, Computer Science Department, West Chester University[EB/OL]. [2016-01-18]. http://www.cs.wcupa.edu/zjiang/matlabindex.htm.
[5] Curricular with the empahsis on secruity, Computer Science Department, West Chester University[EB/OL]. [2016-01-18] http://www.cs.wcupa.edu/isc/curricular.html.
[6] Programcorrectness[EB/OL].[2016-01-18].http://www.bowdoin.edu/~allen/courses/cs260/readings/ch12.pdf.
[7] ACM/IEEE-CS Computer Science Curricula 2013[EB/OL]. [2016-01-18] (2013-12-20). http://www.acm.org/education/CS2013-final-report.pdf.
[8] Schneider D. Essentials of Visual Basic 6.0 Programming[M]. London: Pearson, 1998.
[9] CSC 141 Computer Science I, Computer Science Dept., West Chester University[EB/OL]. [2016-01-18]. http://www.cs.wcupa.edu/zjiang/csc141index.htm.

Dr. Zhen Jiang received BS degree from Shanghai Jiaotong University, China, in 1992, Master degree from Nanjing University, China, in 1998, and PhD degree from Florida Atlantic University, USA, in 2002. Currently, he is associate professor of Computer Science Department at West Chester University of Pennsylvania (WCU) and the directorate of National Security Agency (NSA) certificated Information Security Center at WCU, and an adjunct professor of Temple University (2012-present). His research interests are in the area of information system development and wireless communication. He won the best paper award in the area of protocols and algorithms in the 7th IEEE International Conference on Mobile Ad-hoc and Sensor Systems, 2010. Dr. Jiang is also active in many committees, and holds membership in IEEE and ACM where he is involved in the organization of many of their conferences and workshops.

zjiang@wcupa.edu.

规避编程过程中的不确定性
——一种新的适用于通识课程的教学方法

(Department of Computer Science, Information Security Center, West Chester University, West Chester, PA 19383, USA)

摘 要：针对非计算机专业学生数学和计算机编程薄弱的问题，介绍一个让学生快速掌握决策结构（If-else/If 语句）的教学方法。文章演示如何在 CSC115 这一门通识课中应用该方法，帮助这些基础不够的学生跨越学习过程中的种种不同的疑惑，从而迅速提升学生识习判断语句后的编程技巧。和其他方法（诸如传统的积累方法）不同，该方法更注重循序渐进地对关键技术点进行讲解，以期这些学生能在现有的技术技能基础上理解和执行。这样，学生的学习更加有效和快速。

关键词：认证课程；教学模块和经验；计算机入门教学中的不确定性